Documentation, Variables, Constants, and User Input

Comments

- A comment is text in the code that will be present in the program but will be ignored by Python.
  - That is: comments will not run
- Any line in Python that begins with a ‘#’ will be ignored by Python
- Every high-level programming language will allow you to write text that is not meant to be understood by the computer
- Who/what are comments meant for?

Why Comments?

- Several people can work on a computer program
- Over time computer programs can become very long and complicated
  - Programs can become millions of lines of code long
  - Intentions may change
- If we do not use comments in our code we will have considerable trouble trying to figure out what is going on
- Comments help people that are looking at the code understand it a little better.

Documentation

- At the beginning of your program, always include:
  - Author
  - Revision Date
  - Program name
  - a description of the program

Throughout your program, always comment

- The purpose of each variable
- Explain what is happening in sections of your program
- Not every line needs to be commented but make sure it is clear what is happening throughout each line.

Recall:

- A variable is a location in the computer’s memory where a value can be stored for use later in a program.
- That is, variables can be retrieved and **CHANGED** later in a program

Variable Naming Convention

- Choose descriptive names
  - Should be able to tell what it represents
- Be consistent
  - High_score vs. Highscore
- Follow traditions
  - E.g. start with a lowercase letter, avoid using a '_' as the first letter
- Keep length in check
  - 15 characters or less
- Comment each variable that is created

Constant

- A constant is a variable that does not change during a program

- For example
  - When calculating sales tax in Ontario, the HST is the constant.
  - What is the formula for the area of a circle? What would the constant be?

Literal constant

- The constant value is written right in the program statements

- For example

```
area = 3.14159 * radius * radius
```

Problems with using literal constants

- Suppose we will need to use this constant throughout this program, what are some potential problems?

- Difficult to understand
  - That is, what do these literal constants represent?
- Difficult to maintain
  - If constants were ever to change, we would have to search through the entire program to find them all and change them

Named Constant

- A literal constant which has been given a variable name to represent its value.

e.g.

in the variable declaration section, use

PI = 3.14159

area = PI * radius * radius

- Once declared as a constant, you should not change the value of your constant throughout the program.

- To modify your program, you can only change the value of the constant at the constant declaration.

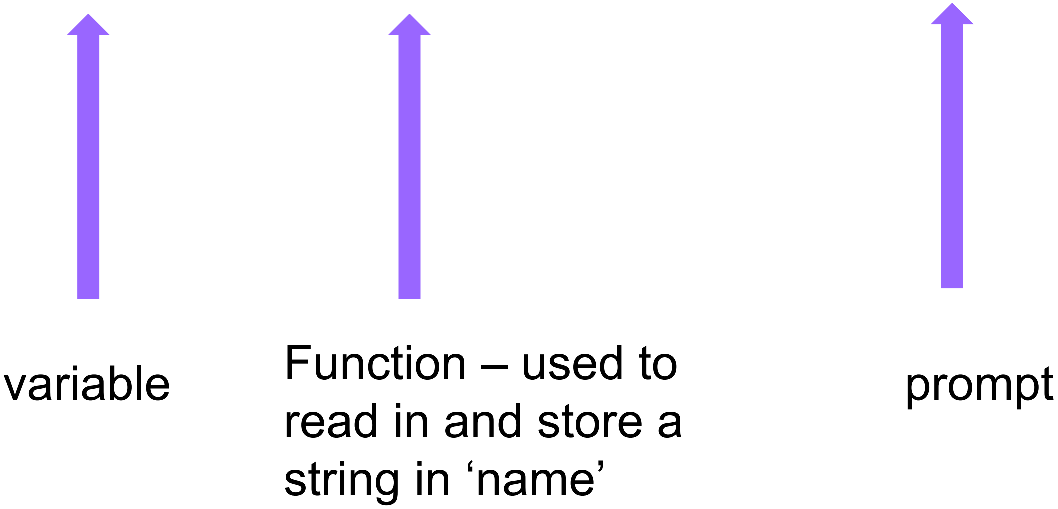
Notes: good programming practice

- Choose meaningful names for constants
- Use capital letters to name the ENTIRE constant
- Declare constants before any other variables
- If you choose two words, separate them with a ‘_’

- Which of the following information would be best stored as a constant?
  - Price of gas
  - The capital of Canada
  - Sales tax
  - Your bank account
  - Your mark today
  - Your final mark

User Input Overview

```
name = input (“Enter your name”)
```



variable

Function – used to read in and store a string in ‘name’

prompt

A prompt should be very specific in what it wants the user to enter!

User Input Overview

```
name = input (“Enter your name”)
```

By default, the variable type being assigned to the variable *name* is of type String.

What if we wanted an integer value instead to perform a mathematical calculation?

int() Function

While *input()* is used to read strings, you can convert the string to a numerical values (both int and float).

```
grade = int(input (“Enter your grade”))
```

Will prompt the user for a grade as a string and convert it to a numerical value of type int

float() function

```
mark = float(input (“Enter a mark”))
```

Will prompt the user for a mark as a string and convert it to a numerical value of type float

str() function

```
mark = str(23.4)
```

Will convert whatever is in the brackets to a String type and then it will get assigned to the variable called mark in the example above.

Will this statement work:

`print (mark + 3)` ?